LA-UR--87-2806

DE87 014765

TITLE: THE PERFORMANCE OF MINISUPERCOMPUTERS: ALLIANT, CONVEX, AND SCS

AUTHOR(S): Harvey J. Wasserman
Margaret L. Simmons
Olaf M. Lubeck

SUBMITTED TO: Vector and Parallel Processing in Computational Science, Liverpool, England, August 25-28, 1987
Parallel Processing

The Performance of Minisupercomputers: Alliant, Convex, and SCS

Harvey J. Wasserman
Margaret L. Simmons
Olaf M. Lubeck

Computing and Communications Division
Los Alamos National Laboratory
Los Alamos, NM 87545
U.S.A.

Abstract. A comparison of the architectures and performance on a set of standard Fortran benchmark codes is made of the Alliant FX, Convex C-1, and SCS-40 minisupercomputers.

1. Introduction

Many definitions of the term "supercomputer" have been given, most of which consider the relative performance and high cost of manufacturing such machines, as well as the unique way in which these machines are programmed and the type of applications on which they are used [6,9,16,21]. Supercomputers can also be described as machines that combine certain advanced architectural features with state-of-the-art electronic componentry. For example, the architectural features associated with such supercomputers as the CRAY X-MP, NEC SX/2, IBM/3090, and the CRAY-2 are (1) various levels of processor parallelism, including multiple CPUs (of which the SX/2 is an exception), vectorization, chaining, and multiple, pipelined functional units; (2) various levels of memory parallelism, including multiple-banked main memory, local memory, cache, and vector registers. The performance of a supercomputer is as much dependent on the ability of the compiler to find parallelism in an algorithm as it is on the hardware speed.

Recently several computers have appeared whose architectures feature many, if not all, of the characteristics of supercomputers, but whose electronic technology uses less expensive and more readily available components [8,10,15]. Such machines have been labeled "minisupercomputers." Currently, this class of computers includes those manufactured by Convex, Alliant, and Scientific Computer Systems (SCS).

The strategy of these minisupercomputer vendors is to produce a machine that may not provide as much absolute performance as the high-end supercomputers but exhibits a comparable or better price-performance ratio. While it is not the goal of this paper to comment on the specific price effectiveness of any particular machine, we believe that a proper evaluation of the minisupercomputers should include price data.

This paper presents a detailed performance comparison of the Convex C-1, Alliant FX/8, and SCS-40 minisupercomputers. While the Convex C-1 is functionally similar to the CRAY-1 supercomputer, the SCS-40 actually emulates a CRAY X-MP/2, running the same instruction set and providing CRAY X-

MP software. The Alliant series of machines are multiprocessors, containing up to eight processors (called computational elements, or CEs) that can run separate processes or cooperate on a single application. The Alliant system also includes a compiler that automatically attempts to partition a code for concurrent execution on several processors. In the first sections of this paper, the single processor performance of the FX/8 is compared with that of the C-1 and the SCS-40, as well as with a Digital Equipment Corporation (DEC) VAX 8600. In a later section, the parallel processing performance of the FX/8 is described.

All measurements were carried out using the standard Los Alamos National Laboratory (LANL) benchmark suite. A recent National Research Council report has characterized supercomputer benchmarks in terms of a useful hierarchy [14]. According to this hierarchy, the codes in the LANL set consist of tests at the levels of hardware demonstration programs, basic routines, and stripped-down applications. The benchmark set has been run on a broad range of both scalar and vector machines [4,7,12,13,18-20].

2. Architectural Considerations

Detailed descriptions of the architectures of the Alliant FX [2,3], Convex C-1 [18], and SCS-40 [17,20] have been published elsewhere. Table I presents a comparison of some functional unit times in the three minisupercomputers, while Table II shows a comparison of some memory features. In both of these tables data are also presented for a CRAY X-MP/48 with a 9.5-ns clock period (CP) for comparison. The three machines discussed in this paper, while all classified as minisupercomputers, are architecturally quite different. For example, Alliant chose to build its processor with moderate vector speed but in a multiprocessor environment, while Convex and SCS designed more raw vector speed into their single processor machines. Additionally, CPU clock periods range from 45 to 170 ns, and maximum vector register lengths range from 32 to 128 64-bit words.

In comparing the minisupercomputers with today's supercomputers we see there are two important differences between these classes of machines, even though precise distinction between the two classes themselves is difficult. (As Lubeck [11] has pointed out, there is a seven-fold variation in CPU clock times within the supercomputer class but only a factor of 4 between the supercomputer and minisupercomputer classes.) Obviously, most differences between the two classes arise from the goal of constructing minisupercomputers at lower cost. The first difference is the method of interconnecting the processor and memory components: supercomputers use direct connections whereas minisupercomputers use busses. The busses in the SCS-40 and the Alliant FX/8 both have cycle times equal to one-half their CPU clock; the Convex C-1 bus, however, runs at twice the CPU cycle time. The second difference is that the Convex C-1 and Alliant FX machines use data cache memory to provide acceptable access times, whereas most supercomputers do not provide caches (the IBM 3090 and NEC SX/2 are exceptions). The minisupercomputers all have nearly the same ratio of memory cycle time to CPU cycle time as the CRAY X-MP/48. Most supercomputer memories are more highly interleaved than the minisupercomputers.

3. Single Processor Benchmark Results

Previous publications have described the important characteristics of the Los Alamos benchmark set [13,20]. All benchmarks were run in 64-bit precision and each code was run in a dedicated environment. In Section 3 all results were from codes requiring only those changes needed to get the codes running; no tuning was done. Most of the Alliant data were collected in March 1987. The Convex data were measured in May 1987, while the benchmark of the SCS-40 took place in December 1986.

3.1 Scalar Performance

Three codes in the benchmark set are non-vectorizable and are thus useful in determining relative scalar performance of the machines. These data are presented in Table III. The order of scalar performance on each of these codes, from fastest to slowest, is SCS-40, Alliant FX, Convex C-1. The SCS-40 is nearly twice as fast as the Alliant. Interestingly, the Alliant is 1.5 to 2 times as fast as the Convex C-1, despite the faster CPU clock on the Convex.

3.2 Basic Vector Operations

We ran two programs, called VECOPS and VECSKIP, that measure the times to execute one million vector operations as a function of vector length. VECOPS makes these measurements using a consecutive memory access scheme, while VECSKIP examines performance with various strided memory references. Results from these codes are listed in Tables IV, V, and VI. The performance advantage of the SCS-40 over the entire range of vector lengths is significant. The faster clock period and higher memory bandwidth of the SCS-40 account for the dramatic differences. Note that these are single processor results for the Alliant. Abu-Sufah and Malony [1] have published a detailed analysis of Alliant FX/8 behavior on our VECOPS program. They measured considerably larger maximum vector rates (than shown in Table V) when the code was executed in vector-concurrent mode, in which individual iterations of a loop are distributed across all eight CEs. However, the FX/8 vector-concurrent rates are still well below the rates we obtained on the SCS-40. The reason for this behavior on the FX/8 is the increased overhead associated with distributing the loop across eight processors and also the decreased memory throughput per CE.

Vector speed as a function of length on most vector processors increases rapidly and eventually reaches an asymptotic rate. The Alliant FX processor reaches its asymptotic rate around vector length 25. For example, in Table V, Alliant vector rates increase by a factor of 2 between lengths 10 and 25, but only increase another 10% from 25 to 1000. As we will see in a later section, this characteristic of the Alliant processor will be important in analyzing parallel processing performance on the FX series.

The time vs. length data may be least-squares fit to a linear model [5] consisting of a startup time, T_o, and an element time, T_e:

$$T = T_o + nT_e \ ,$$

where n is the vector length. However, this model assumes that vectors are loaded directly from main memory into the vector registers. Since this is not true for the Alliant, the derived data are less reliable. The Convex C-1 bypasses cache on vector loads, and we find, for the vector plus scalar operation, T_o = 2800 ns, T_e = 2 CP and, for the vector times vector operation, T_o = 2880 ns, T_e = 3 CP. The Convex, like the CRAY-1, has only one port to memory, capable of one load or store per CP. For the SCS-40, we observe startups of 1463 ns and 1502 ns for these two operations and element times of 1 CP for both. For the chained operation $a(i) = b(i) + s * c(i)$, the SCS-40 is capable of producing two results per clock (and doing two loads/stores per clock), so we observe an element time of 1/2 CP.

It is also instructive to consider vector performance when memory access is not contiguous, shown as the second and third entries in Tables IV-VI. The Convex C-1 does not suffer any performance degradation with stride 23 but it does with stride 8. Although the C-1 bypasses cache in this instance, its memory interleaving is such that only two banks are repeatedly referenced for the stride 8 loop, resulting in significantly lower rates. For the SCS-40, rates calculated from the least-squares fit show that the element time is the same for strided as it is for non-strided access. The startup time on the SCS-40 increases, which decreases the observed rates by about 30% at vector length 10 and 8 % at vector length 1000.

Operations involving irregular memory access schemes, or gather/scatters, constitute the last two operations in Tables IV through VI. The Convex C-1 runs these loops at up to almost twice the rate of either the SCS-40 or the Alliant FX. Although both the Alliant and the Convex vectorize these gather/scatters, both are loading these vectors through cache, and their observed rates are about half what these machines provide on contiguous references. Indirect vector references do not vectorize on the SCS-40.

3.3 Specific Single Processor Benchmark Results

Timing data for the benchmarks are listed in Table VII. For comparison purposes, times measured on a DEC VAX 8600 and one processor of a CRAY X-MP/48 are also given. The X-MP results were measured on a 9.5-ns machine using the CFT77 compiler, while the VAX results were obtained under the VMS operating system using the FORTRAN Version 4.4 compiler. In the specific comments about each code that follow, the results from the Alliant FX/8 are from a single processor.

1. INTMC is an integer Monte Carlo code with virtually no floating point calculations. The Convex C-1 is faster than both the Alliant FX and the SCS-40. The SCS-40 shares the handicap of the CRAY X-MP/24 on this code (previous benchmarks have shown the CRAY machines' weakness on integer arithmetic [7]; the CDC 7600 is two times faster than a CRAY-1) because it does 64-bit integer calculations.

2. FFT, a 64-point transform that is nearly 100% vectorized, runs 4 to 5 times faster on the SCS-40 than it does on the Alliant FX and Convex C-1. This is consistent with the VECOPS data presented above. The algorithm used in FFT is not tuned to any particular machine; vendor-supplied library FFT routines may run much faster.

3. PUSH, the computationally intensive portion of a particle-in-cell hydrodynamics code, contains a few loops with IF statements and makes use of the gather operation. On this code the SCS-40 runs twice as fast as the Convex, which in turn runs twice as fast as the Alliant. This is despite the lack of hardware scatter/gather on the SCS-40.

4. MATRIX performs basic matrix operations on matrices of order 100. Here, the Alliant and Convex times are 3 to 4 times slower than the SCS-40.

5. GAMTEB is a scalar Monte Carlo particle transport code on which the SCS-40 runs twice as fast as the Alliant FX and three times as fast as the Convex C-1.

6. LSS solves a system of linear equations using Gaussian elimination and is almost entirely vectorized. On this code the C-1 runs almost twice as fast as the FX, but the SCS-40 is twice as fast as the C-1. These times are also consistent with the VECOPS data, which showed the Convex to be twice as fast as the Alliant at vector length 100.

7. HYDRO is a Lagrangian hydrodynamics code representative of codes that are a significant part of the Laboratory's workload. The Convex time is about 15% faster than that of the SCS-40, while the Alliant is about twice as slow. HYDRO contains many loops with IF tests that the Convex compiler was able to vectorize but the SCS-40 compiler was not.

4. Parallel Processing on the Alliant FX/8

As mentioned earlier, Alliant provides the first vendor Fortran compiler that attempts automatic parallelization of Fortran programs. The compiler's strategy is to vectorize the innermost loop of a set of nested loops and parallelize the next outer loop (partition it among processors), a mode called "concurrent outer-vector inner" (COVI). When dependencies prevent outer loop concurrency, the inner loop may become vector concurrent. The compiler dependency analysis does not include interprocedural analysis; therefore, function calls in a loop prohibit vectorization and concurrency. Compiler directives can force or prevent concurrency or vectorization in any loop. However, nondeterministic errors are possible with the incorrect use of directives.

Our benchmark of Alliant multiprocessor performance occurred in two stages. The first involved executing codes with no changes. In the second stage we restructured two codes and used compiler directives to force concurrency. The results of both stages are shown in Table VIII, along with the efficiency for eight processors, defined as the ratio of the speedup to 8.

Not surprising to us, there were few places where the original codes could be compiled in COVI mode. This is because most outer loops contain function calls. Thus, in most of the cases where parallelization occurred, the codes were compiled in vector-concurrent mode. The overall performance of vector-concurrent loops represents a tradeoff. In vector-concurrent mode the vector length that each CE executes is 1/P times the original vector length, where P is the number of CEs. As we saw in Section 3.2, vector speed increases with increasing vector length up to about length 25, beyond which it is flat. Partitioning a vector among processors increases multiprocessor utilization at the expense of reducing single processor performance at vector lengths less than 25. Our timing data from HYDRO1 are an example of this tradeoff (in Table VIII HYDRO1 represents HYDRO with no changes). HYDRO is highly vectorized with loops of length 100. The speedup functions for two and four processors are nearly linear because the vector lengths are greater than or equal to 25. However, beyond four processors, the performance curve begins to flatten because of the shorter vectors.

The efficiencies achieved by automatic compiler parallelization (all codes in Table VIII other than HYDRO2) are in the range 14 to 75%. We should remember that these multiprocessor gains come with no extra user effort. Therefore, when comparing the Alliant with the SCS-40 and Convex C-1, it is most fair to use the eight-processor Alliant data. Automatic parallelization by the Alliant compiler has increased the overall performance of the FX/8 so that it is now faster than the Convex C-1 on seven of eight codes.

Significant gains have been made against the SCS-40 also. The strong single processor performance of the SCS-40 is surpassed by the multiprocessor Alliant on two codes (HYDRO and INTMC) and equaled on two others (LSS and MATRIX). The SCS-40 is faster on the remaining five codes, by as little as 40% (PUSH and MONTE) and by as much as a factor of 2 (FFT and EOS). On the codes that represent stripped-down applications in the benchmark hierarchy (PUSH, MONTE, GAMTEB, and HYDRO), the SCS-40 is faster on all but one (HYDRO), which is 60% faster on the Alliant. Stripped-down applications are closer to the full codes in the LANL workload and we weight their performance more heavily.

The inability of the compiler to do interprocedural dependency analysis was the major factor limiting multiprocessing efficiency of the FX/8. Code restructuring is needed to take full advantage of the Alliant model of parallelism, and we attempted this next.

The first code we tried to restructure for concurrency was SCALGAM, a Monte Carlo photon transport code with no vectorization. Theoretically, a high degree of concurrency is possible because each photon is transported independently and there are 200,000 source photons. SCALGAM uses a binary tree of pseudo-random numbers to ensure determinacy. Analysis of the code showed that minor restructuring and a few compiler directives would allow parallelization of the particle loop. In attempting this, however, we encountered non-deterministic errors with more than 5000 source particles. This type of bug occurs all too often in user-defined concurrent programs on any multiprocessor, and we are still trying to debug SCALGAM.

The second code we restructured was HYDRO, a logically rectangular hydrodynamics algorithm with explicit time differencing. This code is a series of nested loops (of depth 2) corresponding to the *k* and *l* lines of the mesh. The Alliant compiler first attempted to run the inner loop (on *k*) in vector-concurrent mode. From previous experience with HYDRO, we know that each outer loop (on *l*) can be executed concurrently, but a function call in the outer loop prevented the compiler from running this loop in parallel. A simple compiler directive should have been all that was needed to run the outer loop concurrently. However, using the directive would have meant that a vector passed (by address) to the function and, modified by it, would become global because the same address would be passed to all processors. A race condition on this argument would thus occur. The solution to this storage model problem was to make the vector local to each outer loop instantiation by rewriting the loop body as a new subroutine that got a separate stack at execution time. The amount of restructuring was large enough that we attempted it for only one loop. The time for HYDRO2 in Table VIII shows the success of this approach when compared with HYDRO1. Better results could have been obtained by applying this solution throughout the code.

5. Conclusions

We have only one code with which we can compare the scalar performance of the DEC VAX 8600 to the minisupercomputers (MONTE). On this code the relative performance of the Alliant, Convex, and SCS to the VAX ranges from 1 to 1.5. However, on our vectorized benchmarks, the minisupercomputers as a group ran 2 to 20 times faster than the VAX. On most of the codes, the SCS-40 showed the biggest improvement over the VAX.

The SCS-40 and Alliant FX/8 both out-perform the Convex C-1, except on HYDRO, where the C-1, because its compiler is better at vectorization, performs about 15% better than the SCS-40. The SCS-40 is 1.5 to 5 times as fast as a single processor of the Alliant FX. However, in comparing the Alliant FX with the SCS-40 we note, again, that it is most appropriate to compare the unrestructured eight-processor Alliant results with the SCS-40 because both represent the maximum performance from each machine with no user intervention. Performance results for the SCS-40 and FX/8 are mixed. Placing more emphasis on our stripped-down application benchmarks, the SCS-40 maintains a performance edge. In general, we believe the SCS-40 has an advantage on our codes because of its faster CPU clock and higher memory bandwidth, even though we believe its compiler technology is inferior to that of Alliant and Convex.

Our codes showed as much as a speedup of four from a parallelization in vector-concurrent mode. On the other hand, to extract significantly more parallelism from our codes, interprocedural analysis is necessary. This is a difficult problem, and to the best of our knowledge, no computer vendor currently offers such a facility in a production level compiler.

Finally, we emphasize that the benchmark codes used in this study are intended to represent only the Los Alamos workload, and caution should be used in comparing the results with those based on other workloads.

6. Acknowledgements

This work was performed under the auspices of the U.S Department of Energy. We thank Gayle Johnson of Alliant, Ron Gray of Convex, and Chuck Niggly of SCS for their assistance during the benchmarks. We thank Lawrence Pratt of LANL Group CLS-2 for allowing us to use the Convex C-1.

7. References

1. W. Abu-Sufah and A. D. Malony, Vector Processing on the Alliant FX/8 Multiprocessor, *Proc. IEEE Intl. Conf. Parallel Processing* (1986) 559-570.

2. W. Abu-Sufah and A. D. Malong, Experiences with the Alliant FX/8 Mini-Supercomputer, *CSRD Report #570, Center for Supercomputing Research and Development, University of Illinois, 1986.*

3. Alliant FX/Series Product Summary, Alliant Computer Systems Corporation, Littleton, Massachusetts (1986).

4. R. G. Brickner, H. J. Wasserman, A. H. Hayes, and J. W. Moore, Benchmarking the IBM 3090 with Vector Facility, *Los Alamos National Laboratory Unclassified Release LAUR-86-3300* (1986).

5. I. Y. Bucher and J. W. Moore, Comparative Performance Evaluation of Two Supercomputers: CDC Cyber-205 and CRAY-1, *Los Alamos National Laboratory Unclassified Release LAUR-81-1977* (1981).

6. M. D. Ercegovac and T. Lang, General Approaches for Achieving High Speed Computations, in: S. Fernbach, ed., *Supercomputers Class VI Systems, Hardware and Software* (North Holland, Amsterdam, 1986) 1-28.

7. J. H. Griffin and M. L. Simmons, Los Alamos National Laboratory Computer Benchmarking 1986, *Los Alamos National Laboratory Unclassified Report LA-10151-MS* (1984).

8. S. Lackey, J. Veres, and M. Ziegler, Supercomputer Expands Parallel Processing Options, *Computer Design* (1985) 76-81.

9. N. Lincoln, Technology and Design Tradeoffs in the Creation of a Modern Supercomputer, in: S. Fernbach, ed., *Supercomputers Class VI Systems, Hardware and Software* (North Holland, Amsterdam, 1986) 83-111.

10. J. R. Lineback, CMOS Gates Key to 'Affordable' Supercomputer, *Electronics Week* (1984), 17-20.

11. O. Lubeck, Supercomputer Benchmarks: Theory, Practice, and Results, in: M. Yovits, ed., *Advances in Computers* (North Holland, Amsterdam), to appear 1988.

12. O. Lubeck, J. Moore, and R. Mendez, A Benchmark Comparison of Three Supercomputers: Fujitsu VP-200, Hitachi S810/20, and CRAY X-MP/2, *IEEE Computer* 18 (1985) 10-29.

13. O. Lubeck, J. Moore, and R. Mendez, The Performance of the NEC SX/2 and CRAY X-MP Supercomputers, Los Alamos National Laboratory Unclassified Release LAUR-87-227 (1987).

14. An Agenda for Improved Evaluation of Supercomputer Performance, National Research Council (National Academy Press, Washington D.C., 1986).

15. S. Ohr, Computer Scores with Cray Compatibility, *Electronic Design* (1985) 61-62.

16. J. P. Riganati and P. B. Schneck, Supercomputing, *Computer* 17 (1984) 97-113.

17. SCS-40 Architecture Reference Manual, Scientific Computer Systems Corporation, *SCS Document Number 950000001-001A*, San Diego, California (1986).

18. M. L. Simmons and O. Lubeck, Benchmark of the Convex C-1 Mini Supercomputer, *Los Alamos National Laboratory Unclassified Release LAUR-86-2890* (1986).

19. M. L. Simmons and H. J. Wasserman, Los Alamos National Laboratory Computer Benchmarking 1986, *Los Alamos National Laboratory Report LA-10898-MS* (1987).

20. H. J. Wasserman, M. L. Simmons, and A. H. Hayes, A Benchmark of the SCS-40 Computer: A Mini Supercomputer Compatible with the CRAY X-MP/24, *Los Alamos National Laboratory Unclassified Release LAUR-87-659* (1987).

21. J. Worlton, Supercomputers: Past, Present and Future, presented at Second SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, Virginia, November, 1984.

Table I. Representative Functional Unit Operation Times

	Floating Point Add	*Floating Point Multiply*	*Floating Point Divide*	*Integer Add*
Alliant FX/8	510 ns (3 CP)	510 ns (3 CP)	2720 ns (16 CP)	680 ns (4 CP)
Convex C-1	300 ns (3 CP)	400 ns (4 CP)	3300 ns (33 CP)	100 ns (1 CP)
SCS-40	135 ns (3 CP)	135 ns (3 CP)	270 ns (6 CP)	90 ns (2 CP)
CRAY X-MP/48	57 ns (6 CP)	66.5 ns (7 CP)	133 ns (14 CP)	28.5 ns (3 CP)

Table II. Important Memory Characteristics

	Cycle Time	*Maximum Vector Transfer Rate*	*Vector Access Time*	*Bus Throughput*	*Number of Banks*
Alliant FX/8		2 word/CP		188 Mbyte/s	8
Convex C-1	4 CP 400 ns	4 word/CP	3 CP/8 CP* 300 ns/800 ns	80 Mbyte/s	16
SCS-40	5 CP 225 ns	4 word/CP	5 CP 225 ns	1 Gbyte/s	16
CRAY X-MP/48	4 CP 38 ns	3 word/CP	17 CP 161.5 ns	-	64

* Vector in cache/vector not in cache.

Table III. Execution Times (in Seconds) for Scalar Codes

Program Name	*Alliant FX*	*SCS-40*	*Convex C-1*
GAMTEB	41.0	21.0	61.0
EOS	157.	77.8	410.
MONTE	19.0	13.6	21.0

Table IV. Rates (MFLOPS) on the Convex C-1 for Selected Vector Operations as a Function of Vector Length

Operation	*10*	*25*	*50*	*100*	*200*	*1000*
a(i) = b(i) + s	1.5	2.5	3.0	3.5	3.7	4.0
a(i) = b(i) + s (i=1,n,23)	1.1	2.1	2.9	3.6	3.8	4.0
a(i) = b(i) + s (i=1,n,8)	0.7	1.0	1.1	1.2	1.2	1.2
a(i) = b(i) * c(i)	1.1	1.8	2.2	2.5	2.6	2.8
a(i) = b(i) + s * c(i)	2.4	3.5	4.4	5.0	5.2	5.5
a(i) = b(i) * c(i) + d(i) * e(i)	2.3	3.2	4.0	4.5	4.9	4.9
a(i) = b(j(i)) + s	0.9	1.5	2.0	2.1	2.0	2.0
a(j(i)) = b(i) * c(i)	0.8	1.1	1.4	1.6	1.6	1.7

Table V. Rates (MFLOPS) on the Alliant FX for Selected Vector Operations as a Function of Vector Length

Operation	*10*	*25*	*50*	*100*	*200*	*1000*
a(i) = b(i) + s	1.2	1.9	2.0	2.1	2.2	2.3
a(i) = b(i) + s (i=1,n,23)	0.7	1.2	1.6	1.8	2.0	2.0
a(i) = b(i) + s (i=1,n,8)	0.7	1.3	1.6	1.8	2.0	2.0
a(i) = b(i) * c(i)	1.0	1.4	1.5	1.5	1.6	1.6
a(i) = b(i) + s * c(i)	1.6	2.5	2.7	2.8	2.9	3.1
a(i) = b(i) * c(i) + d(i) * e(i)	1.5	2.0	2.0	2.1	2.	2.2
a(i) = b(j(i)) + s	0.7	1.0	1.1	1.1	1.1	1.2
a(j(i)) = b(i) * c(i)	0.6	0.9	1.0	1.1	1.1	1.2

Table VI. Rates (MFLOPS) on the SCS-40 for Selected Vector Operations as a Function of Vector Length

Operation	*10*	*25*	*50*	*100*	*200*	*1000*
a(i) = b(i) + s	3.6	9.0	15.3	17.3	17.8	20.2
a(i) = b(i) + s (i=1,n,23)	2.6	6.4	10.9	13.8	15.3	18.6
a(i) = b(i) + s (i=1,n,8)	--	--	--	--	--	--
a(i) = b(i) * c(i)	3.5	8.7	13.8	15.5	16.0	18.2
a(i) = b(i) + s * c(i)	6.6	16.4	27.5	30.7	31.9	36.4
a(i) = b(i) * c(i) + d(i) * e(i)	8.4	17.8	24.6	25.8	26.2	27.3
a(i) = b(j(i)) + s	0.9	1.0	1.0	1.0	0.9	1.0
a(j(i)) = b(i) * c(i)	1.0	1.1	1.2	1.2	1.2	1.2

Table VII. Benchmark Execution Times (in Seconds) for the Alliant FX, SCS 40, CONVEX C-1, DEC VAX 8600, and CRAY X-MP/48*

Program Name	*Alliant FX*	*SCS-40*	*Convex C-1*	*DEC VAX 8600*	*CRAY X-MP/48*
FFT	82.0	21.0	97.0	222.2	4.2
MATRIX	953.1	167.0	529.0	-	43.2
LSS	120.0	28.0	65.0	462.2	7.4
GAMTEB	41.0	21.0	61.0	-	5.9
INTMC	322.0	219.0	83.0	186.9	13.3
PUSH	190.0	39.0	87.3	332.0	4.8
MONTE	19.0	13.6	21.0	20.2	2.0
HYDRO	771.0	347.8	298.2	-	17.2
EOS	157.	77.8	410.	155.2**	21.5

* X-MP and Alliant FX times are from a single processor.

** Thirty-two-bit precision for this code.

Table VIII. Execution Times (in seconds) on 1 - 8 Processors of the Alliant FX/8

Program Name	*One Processor*	*Two Processor*	*Four Processor*	*Eight Processor*	*Efficiency for Eight Processors*
FFT	82.0	59.0	48.0	40.0	.25
MATRIX	953.1	478.6	263.7	156.5	.75
LSS	120.	64.	38.	26.	.58
GAMTEB	41.0	38.0	36.0	36.0	.14
INTMC	322.0	215.0	160.0	153.0	.26
PUSH	190	113	74	56	.42
MONTE	19	19	19	19	-
HYDRO.1	771	452	290	208	.46
HYDRO.2	771	415	248	163	.59
EOS	157	155	154	154	-